Software Configuration Management

Trabalho Prático 2
ESII 2021/22

Rui Neto 8200321
Simão Santos 8200322

Topics

1. SCM Management 2
1.1 SCM Roles and Responsibilities 2
2. CM Activities 3
2.1 Configuration Identification 3
2.2 Configuration Control 3
2.3. Configuration Status and Accounting 3
3. CM in the Software Development Life Cycle 4
4. Tools and Methodology 5

1. SCM Management

1.1 SCM Roles and Responsibilities

1.1.1 Dev Team

- The Dev Team members are Rui Neto and Simão Santos.
- When these developers would like to make changes to any code or documentation, they must create a GitLab Issue and get approval from another Developer.
- They must use the change request forms provided by the Change Request Template Issue.

1.1.2 CM Team (Configuration Management team)

- The CM team consists of: Rui Neto and Simão Santos.
- The team is responsible for maintaining all official project documentation, code, and underlying software for the duration of the project.

1.1.2.1 CM Leaders

- CM leaders are Rui Neto an Simão Santos.
- The CM Leaders composes the CM Plan, directs the establishment of the CCB, directs the establishment and directs the establishment of coding conventions.
- The CM Leader will also run code review meetings to keep them on-task and on-time.
- More complex change requests that result in a commit/merge request necessitate a code review run by another CM Leader.

1.1.3 Configuration Control Board (CCB)

- This team is responsible for approving changes to all software and documentation described earlier for the duration of the project. It is the responsibility of the CCB to direct developers as to how they are involved in change approval.

1.1.3.1 CCB Leaders

- CCB Leaders: Rui Neto and Simão Santos.
- Runs CCB meetings and sends out notices for meetings to project team members. The CCB Leaders must share the change request in call meetings to review/approve the change requests, discontinue efforts on specific change requests, and establishes change priorities.

2. CM Activities

This section shows how the project's versions will be controlled.

2.1 Configuration Identification

- For the versions control we will use branches. Starting with the master branch which has the functional part of the system, followed by the Dev branch where the merge problems (that can exist or not) will be verified.
- The commit message must has the software change.
- A change must be asked using an issue with a change request title in GitLab.

2.2 Configuration Control

1. Submit an Issue on GitLab identifying the change request (following the template);
2. Fill all the tasks in the in the 'Change Technician checklist';
3. CCB Leader or Developer must review and decide if the change request is possible and makes sense;
4. Pre-condition -> The third step is done and the change request is alright. Fill all the tasks in the 'Change Reviewer checklist';
5. Add the changes to the local branch.
6. Commit and push to the 'private' branch of the online repository.
7. On GitLab create a merge request that later must be verified by a Developer or a CM Leader.
8. After the permission and verification, check if the merge has problems. If not, follow the next steps.
9. At some point merge the functional tools to the master branch.
10. Close the issue.

2.3. Configuration Status and Accounting

- Agile development

Specification, design, implementation and testing are interleaved and the outputs from the development process are decided through a process of negotiation during the software development process.

3. CM in the Software Development Life Cycle

- Our Software Development Life Cycle will be nothing more than:

  Planning;

  Requirements;

  Build;

  Document;

  Test;

  Deploy;

  Maintain the process.

- The SCM plan supports all these activities by following some subprocesses represented bellow:

  Documentation;

  Configuration Compliance;

  Quality Assurance;

  Verification;

  Validation;

  Joint Review;

  Problems Solving;

  Usability.

4. Tools and Methodology

- Tools:

Java – Linguagem de programação usada para o desenvolvimento do motor de pesquisa e especificação de testes;

Gradle – Ferramenta de configuração e automação de builds de software;

jUnit – Framework para codificação dos testes especificados;

Jacoco – para análise de cobertura do código, efetuando a contagem da percentagem de cobertura dos testes, face ao total de instruções, ramos, linhas de código, métodos e classes.

PMD – Ferramenta para análise estática de código e checktyle;

Gitlab - controlo de versões e organização do projeto;

- The methodology that will be used is *SCRUM*

We use boards that have “open”, “in progress”, and other columns.

We use a backlog to plan which features will be added to the sprint. Estimate our tasks and track the results of the sprint.

Schedule a sprint and add a goal so the whole team is aware of it. When the sprint is finished, we add unresolved issues to a new sprint.

Get an overview of our team activities using the dashboards and workers activities and repeat the process constantly.